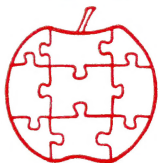


Apple

\$1.50



Assembly

Line

Volume 3 -- Issue 1

October, 1982

In This Issue...

Catalog Arranger	2
So You Never Need Macros!	17
Converting ToolKit Source to S-C	21
Correction to Bob's Fast Screen Scroll	28
Using USR for a WEEK	30
Automatic CATALOG for the Language Card Version	31
Another Lower Case Patch for S-C Macro	32
Writing for AAL	32

This issue of AAL is late. No sooner do I warn you of one magazine that is behind in their publication schedule, than I get behind myself! We plan to catch up with the next issue.

I just returned from 8 days in California. Some of you know that I am on the board of directors of the International Apple Core. After the board meeting I contacted a few long-time customers. I also attended the San Francisco Apple Corps Swap Meet.

I looked up Peter Meyer, author of SDS's "Routine Machine"; together we had dinner at the home of Pat Caffrey, co-author of "Doubletime Printer". Peter is now working on the fourth volume of additional Applesoft-extenders for his Routine Machine. I also spent two half-days with Henry Spragens, well known for his early contributions to Apple graphics lore. He bought his (first) Apple long ago in Kentucky, where he was one of the original members of LAUGHS (Louisville Apple User's Group for Hardware and Software!). Now he works at Beck-Tech in Berkeley, doing exotic things in the world of synthesized video graphics with the Apple and other machines.

A few weeks earlier I spent the afternoon with DeWayne Van Hoozer in Houston, at the HAAUG meeting (you're right, it is pronounced "hog"!). DeWayne's Genasys project is nearing the publication stage, so you'll probably be hearing more about it soon. I am looking forward to some more time in Houston around Halloween, at the AppleFest there. Look me up if you are there, in or near the International Apple Core booth.

Catalog ArrangerBill Morgan

We all have the problem: a disk starts getting full, we delete some files to make space, and our new files (from our latest project) end up scattered all through the catalog. A disk that has been used for a few months ends up with a thoroughly shuffled catalog.

There are programs available to alphabetize a catalog, but that's not always what I want to do. I want HELLO at the beginning, utilities next (assembler, text editor, ES-CAPE, disk zap, etc.), then various projects. The files for each project should all be grouped together, with the current job at the end of the catalog.

I decided that what I want to be able to do is to "pick up" one entry in the catalog, move it to exactly where I want it, put it down, then go get another one and put that one in its place, and so on. Here's my program to do just that.

Using Catalog Arranger

First BLOAD CATALOG ARRANGER, then insert the disk you want to modify. When you type CALL 2051 (or 803G from the monitor) the disk will spin for a little while as the catalog is read into a sort of string array. The first 22 entries in the catalog will then be displayed, with the first entry shown in inverse. You may notice that deleted files are also displayed, with a minus sign before the file type and a stray inverse character out at the end of the file name. Control characters are also displayed in inverse.

The inverted entry is a cursor showing the "active entry". If you press the arrow keys, this cursor will move up and down the display. When the cursor reaches the center of the screen, it will stop moving and the display will scroll up and down around it.

When you have the cursor on an item you wish to move, press RETURN. The word "MOVING" will appear in inverse in the lower left corner of the screen. When this "moving flag" is on, the entry in the cursor will be carried wherever the cursor goes. When it reaches the place where you want to put it, press RETURN again. The moving flag will disappear and that entry will stay where you just put it.

There are a couple of other commands as well. Pressing the "B" key moves the cursor to the beginning of the catalog, and the "E" key moves it to the end. If the moving flag is on, the item in the cursor will be carried right along. There is also an "R" command, to read in a new catalog. This is useful if you want to reread the current catalog and start all over again, or to move on to another disk.

When you have the catalog arranged just the way you want it, press the "W" key to write the revised catalog onto the disk. Press ESC when you want to exit the program.

APPLE PERIPHERALS ARE OUR ONLY BUSINESS

TIME II

THE MOST POWERFUL, EASIEST TO USE CLOCK FOR YOUR APPLE

- Time in hours, minutes and seconds.
- Date with year, month, day of week and leap year.
- Will enhance programs for accounting, time and energy management, remote control of appliances, laboratory analysis, process control, and more.
- 24-hour military format or 12-hour with AM/PM indication.
- User selectable interrupts permit foreground/background operation of two programs simultaneously.
- Crystal controlled for .0005% accuracy.
- Easy programming in basic.
- On board battery backup power for over four months power off operation (battery charges when Apple is on).



- Twenty-seven page operating manual included with many examples of programs to use with your Apple in any configuration.
- Includes disk containing a DOS Dater and many other time oriented utilities plus over 25 user contributed programs at no extra cost.

PRICE \$129.00

SUPER MUSIC SYNTHESIZER



- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo and boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.

- We give you lots of software. In addition to Compose and Play programs, the disk is filled with songs ready to run.
- Easy to program in basic to generate complex sound effects.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Envelope control.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all the features of this board. Their software sounds the same in our synthesizer.)
- Automatic shutdown on power-up or if reset is pushed.
- Many many more features.

PRICE \$159.00

ANALOG TO DIGITAL CONVERTER

- 8 Channels
- 8 Bit Resolution
- On Board Memory
- Ratiometric Capability
- Fast Conversion (.078 ms per channel)
- Eliminates The Need To Wait For A/D Conversion (just PEEK at data)
- A/D Process Totally Transparent to Apple (looks like memory)

The analog to digital conversion takes place on a continuous, channel sequencing basis. Data is automatically transferred to on board memory at the end of each conversion. No A/D converter could be easier to use.

Our A/D board comes standard with 0, 10V full scale inputs. These inputs can be changed by the user to 0, -10V, or -5V, +5V or other ranges as needed.

The user connector has +12 and -12 volts on it so you can power your sensors. (These power sources can be turned off with on board dip switch).

Accuracy 0.3% Input Resistance 20K Ohms Typ
A few applications may include the monitoring of • flow • temperature • humidity • wind speed • wind direction • light intensity • pressure • RPM • soil moisture and many more.

PRICE \$129.00

DIGITAL INPUT/OUTPUT BOARD

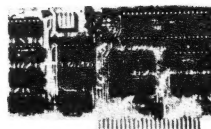
- Provides 8 buffered outputs to a standard 16 pin socket for standard dip ribbon cable connection.
- Power up reset assures that all outputs are off when your Apple is first turned on.
- Features 8 inputs that can be driven from TTL logic or any 5 volt source.
- Your inputs can be anything from high speed logic to simple switches

- Very simple to program, just PEEK at the data.
- 4 other outputs are also provided. User 1, reset, interrupt request, non-maskable interrupt.
- Now on one card, you can have 8 digital outputs and 8 digital inputs each with its own connector. The super input/output board is your best choice for any control application.

PRICE \$62.00

Z-80 CARD

- TOTALLY compatible with all CP/M software.
- Executes the full Z-80 and 8080 instruction set.
- Allows you to run your Apple CP/M based programs.
- Does EVERYTHING the other Z-80 boards do, plus supports Z80 Interrupts
- Hardware and software settable switch options.
- An on-card PROM eliminates many I.C.'s for a cooler, less power consuming board.
- Complete documentation included. (user must furnish)



PRICE \$139.00

Since our inception, Applied Engineering has continually expanded its line of Apple peripherals bringing you easy-to-use designs.

We are the innovators not the imitators. Utilizing state-of-the-art technologies, Applied Engineering is continually improving its products. The above represents our most recent development. Applied Engineering offers you the highest quality peripherals at the lowest possible price.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a one year warranty.

All Orders Shipped Same Day.
Texas Residents Add 5% Sales Tax.
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to
APPLIED ENGINEERING
P.O. Box 470301
Dallas, TX 75247

See Your Dealer
or Call (214) 492-2027
7 Days a Week
Master Card & Visa Welcome

Catalog Organization

If you are familiar with the internal structure of an Apple DOS catalog, you can skip ahead to the section labelled "How Catalog Arranger Works".

The first step in reading a disk catalog is to read the VTOC (Volume Table of Contents), which is always located at track \$11, sector 0. The second and third bytes in the VTOC (offsets 1 and 2) contain the track and sector of the start of the catalog. On a standard DOS 3.3 disk these always point to track \$11, sector \$0F, and the rest of the catalog is always on track \$11. These locations can be changed, however. For example, some programs to convert DOS 3.2 disks to DOS 3.3 leave the first catalog sector at track \$11, sector \$0C. Therefore, it is safest to follow the pointers rather than assuming that the catalog will always be in its usual place.

In a catalog sector, the first byte is not used. The second and third bytes point to the next track and sector of the catalog. If the track byte is zero, it means that there is no next sector, this is the end of the catalog. The fourth through the eleventh bytes are not used. The actual catalog information starts at the twelfth byte of the sector (offset \$B) and fills the rest of the sector. Each catalog entry takes 35 bytes, so there are 7 entries in each sector.

The first two bytes of an entry contain the track and sector of the file's track/sector list. If the first byte is \$FF, the file has been deleted. In that case, the track number has been moved to the end of the file name. If the first byte is zero, this entry has never been used, and we are past the end of the catalog.

The third byte tells the file type and whether the file is locked. Here are the type codes:

- 00 -- TEXT file
- 01 -- INTEGER BASIC program
- 02 -- APPLESOFT BASIC program
- 04 -- BINARY file
- 08 -- S file -- not used
- 10 -- R file -- DOS Toolkit relocatable object file
- 20 -- A file -- Lazer Pascal source file
- 40 -- B file -- Lisa assembler source file

If the file is locked, \$80 is added to the type code.

The next 30 bytes are the file name. If the name is less than 30 characters long, it is filled out with spaces (ASCII \$A0).

The last two bytes are the length of the file, in sectors. This is expressed as first low byte, then high byte.

Limitations and Additional Features

There are a few points that need more work:

Disk error handling. The program just prints "I/O ERROR" and stops. It needs a real error handler.

This program will handle catalogs with up to 127 entries. A standard disk has no more than 105, but some catalogs are modified to have more.

I plan to add several more commands to Catalog Arranger:

Alphabetic sort. This would be useful too, maybe just from the cursor to the end of the catalog. That would keep the utilities in place at the beginning.

Sort by file type.

Delete and undelete files.

Move deleted files to the end.

Rename files by editing the file name in the cursor. That would be a lot easier than typing entire file names twice, as RENAME requires.

Display and allow changing the values of SLOT and DRIVE.

Display the value of ACTIVE.ELEMENT and NUMBER.OF.ENTRIES, and maybe free sectors on the bottom line.

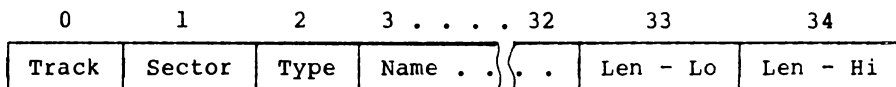
Write the catalog out to the disk as a text file.

Adding many of these features would also require reworking the command structure (which wouldn't hurt anyway!)

Here is a summary of the commands:

```
B -- Move the cursor to the beginning of the catalog.
E -- Move the cursor to the end of the catalog.
R -- Read the catalog from the disk.
W -- Write the catalog to the disk.
--> -- Move the cursor down one item.
<-- -- Move the cursor up one item.
RETURN -- Toggle the moving flag on or off.
ESC -- Exit the program.
```

Here's a diagram:



You can find more information on the structure of the catalog in the DOS Manual on pages 129-134 or in the book *Beneath Apple DOS* on pages 4-4 through 4-7. It is impossible to recommend *Beneath Apple DOS* too highly; if you have any interest in the internals of DOS, get that book.

How Catalog Arranger Works

After initialization, the program builds a table of pointers into the storage area. We can build this table in advance because we know that all entries will be the same length. The catalog is then read into the array, each entry being placed according to the next pointer in the table. The end of the table is then marked with two zero bytes after the last element used.

The next step is to display the entries in the array. The display routine starts by checking `ACTIVE.ELEMENT` to see whether to start the display with the first element, or somewhere in the middle. It then scans up the table, displaying each catalog entry and inverting the one corresponding to `ACTIVE.ELEMENT`. The routine that actually displays each line borrows a couple of subroutines in DOS to decode the file type and display the file size.

When `MOVING.FLAG` is off, the arrow, B, and E commands simply change the value of `ACTIVE.ELEMENT`. When `MOVING.FLAG` is on, the arrows swap entries in the table up or down, and B and E repeatedly swap entries to move `ACTIVE.ELEMENT` to the beginning or end.

When writing the catalog back onto the disk we have to be careful to put the catalog sectors back in the same place they came from, since we can't assume that the catalog came from track \$11. We do this by reading the first catalog sector into the buffer, scanning up the pointer table and moving the indicated entries into the catalog buffer, and then writing the buffer to the same disk sector it came from. We then get the track and sector pointers (which haven't been changed) from the buffer and use them to read the next sector. This whole process ends when we run out of entries in the pointer table.

CAN YOU PROGRAM YOUR SERIAL CARD? NOW YOU CAN.

The SPI card gives your APPLE II* computer a new dimension to serial I/O: firmware on RAM. This allows you to adapt the SPI to virtually any RS-232 serial device. It gives you total control.

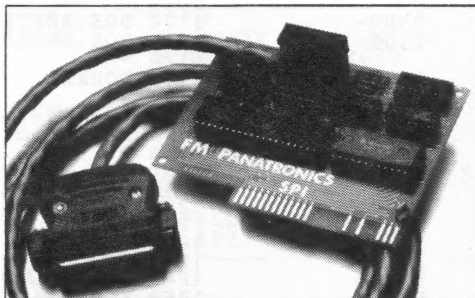
BSAVE your SPI programs on disk. Then BLOAD the one you want directly into the SPI RAM. Anytime. Even on boot-up.

Control characters select the various features and modes: dumb terminal, remote computer, and printer interface. However, if you like to customize your firmware or have an unusual device to operate, well-documented source listings are included.

Now the frosting. The SPI RAM can be used to store machine language routines and utilities safely out of main memory. It's a new way to use peripheral memory.

The SPI costs only \$99.50 including 5' cable and documentation. For longer cable lengths add \$1.00 per foot.

*APPLE is a trade mark of Apple Computer Inc.



YOU HAVE TOTAL CONTROL

- Firmware in 2K x 8 static RAM.
- Control CTS, DSR, DTR, RTS, RLSD.
- Control baud rate, word length, parity, etc.
- Programs supplied for: RAM test, UART, terminal mode, remote mode, printer mode.
- 1 year warranty.

FM PANATRONICS Corp.

Box 1088 Stone Mountain, GA 30086

PRICE: \$99.50*
POST PAID

*Georgia residents please add 3% state sales tax and local sales tax if applicable. Postage prepaid inside continental U.S. (outside U.S. add \$5.00). All payments must be made in U.S. dollars. Please allow 4 to 6 weeks for delivery.

☐ Check or Money Order

☐ VISA

☐ MasterCard

CARD NO.

EXPIRATION DATE

AUTHORIZED SIGNATURE

NAME (please print)

ADDRESS

CITY

STATE

ZIP

FM PANATRONICS, Box 1088, Stone Mountain, GA 30086

A2

```

1010 *-----
1020 .OR $803
1030 * .TF CATALOG ARRANGER
1040 *-----
0000- 1050 POINTER .EQ 0
1060 *
0025- 1070 MON.CV .EQ $25
0048- 1080 PREG .EQ $48
1090 *
03D0- 1100 DOS.RESTART .EQ $3D0
03D9- 1110 DOS.RWTS .EQ $3D9
1120 *
07D0- 1130 CORNER .EQ $7D0
1140 *
C000- 1150 KEYBOARD .EQ $C000
C010- 1160 KEYSTROBE .EQ $C010
1170 *
AE42- 1180 DOS.SIZEOUT .EQ $AE42
A702- 1190 DOS.PRNTERR .EQ $A702
B3A7- 1200 DOS.TYPTABL .EQ $B3A7
B7E8- 1210 IOB .EQ $B7E8
B7E9- 1220 IOB.SLOT .EQ $B7E9
B7EA- 1230 IOB.DRIVE .EQ $B7EA
B7EB- 1240 IOB.VOLUME .EQ $B7EB
B7EC- 1250 IOB.TRACK .EQ $B7EC
B7ED- 1260 IOB.SECTOR .EQ $B7ED
B7F0- 1270 IOB.BUFFER .EQ $B7F0,F1
B7F4- 1280 IOB.COMMAND .EQ $B7F4
B7F5- 1290 IOB.ERROR .EQ $B7F5
B7F7- 1300 IOB.OSLOT .EQ $B7F7
B7F8- 1310 IOB.ODRIVE .EQ $B7F8
1320 *
1330 * MONITOR CALLS
1340 *
FC22- 1350 MON.VTAB .EQ $FC22
FC42- 1360 MON.CLREOP .EQ $FC42
FC58- 1370 MON.HOME .EQ $FC58
FDDA- 1380 MON.PRBYTE .EQ $FDDA
FDF0- 1390 MON.COUT1 .EQ $FDF0
FE80- 1400 MON.SETINV .EQ $FE80
FE84- 1410 MON.SETNORM .EQ $FE84
1420 *
1430 * SYMBOLIC CONSTANTS
1440 *
0000- 1450 ZERO .EQ 0
0001- 1460 READ .EQ 1
0002- 1470 WRITE .EQ 2
0016- 1480 LINE.COUNT .EQ 22
0023- 1490 ENTRY.LENGTH .EQ 35
008D- 1500 RETURN .EQ $8D
00A0- 1510 SPACE .EQ $A0
1520 *-----
1530 SETUP
0803- AD F7 B7 1540 LDA IOB.OSLOT SET SLOT AND
0806- 8D 2D 0B 1550 STA SLOT DRIVE TO WHERE
0809- AD F8 B7 1560 LDA IOB.ODRIVE WE CAME FROM
080C- 8D 2E 0B 1570 STA DRIVE
1580
080F- A9 00 1590 LDA #ZERO INITIALIZE
0811- 8D 2F 0B 1600 STA VOLUME VARIABLES
0814- 8D 35 0B 1610 STA NUMBER.OF.ELEMENTS
0817- 8D 37 0B 1620 STA MOVING.FLAG
081A- A9 FF 1630 LDA #$FF
081C- 8D 36 0B 1640 STA ACTIVE.ELEMENT
1650
081F- 20 9C 0A 1660 JSR BUILD.ARRAY.TABLE
0822- 20 BD 08 1670 JSR READ.CATALOG
0825- 20 58 FC 1680 JSR MON.HOME
1690
*-----
0828- 20 97 09 1700 DISPLAY.AND.READ.KEY
1710 JSR DISPLAY.ARRAY
1720
082B- AD 00 C0 1730 .1 LDA KEYBOARD
082E- 10 FB 1740 BPL .1
0830- 8D 10 C0 1750 STA KEYSTROBE
0833- C9 95 1760 CMP #$95 -->
0835- F0 1F 1770 BEQ HANDLE.RIGHT.ARROW
0837- C9 88 1780 CMP #$88 <--
0839- F0 2E 1790 BEQ HANDLE.LEFT.ARROW
083B- C9 9B 1800 CMP #$9B ESC
083D- F0 78 1810 BEQ HANDLE.ESC

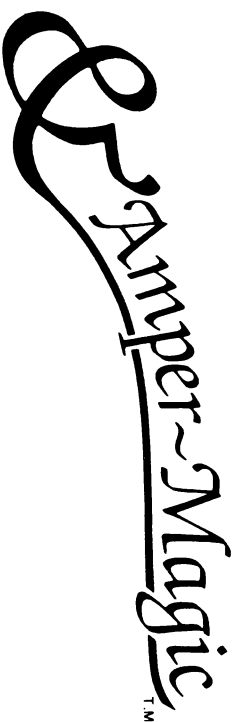
```



```

083F- C9 8D 1820 CMP #RETURN
0841- F0 69 1830 BEQ HANDLE.RETURN
0843- C9 C2 1840 CMP #C2
0845- F0 35 1850 BEQ HANDLE.B BEGINNING
0847- C9 C5 1860 CMP #C5 E
0849- F0 46 1870 BEQ HANDLE.E END
084B- C9 D2 1880 CMP #D2 R
084D- F0 6B 1890 BEQ HANDLE.R READ CATALOG
084F- C9 D7 1900 CMP #D7 W
0851- F0 53 1910 BEQ HANDLE.W WRITE CATALOG
0853- 4C 2B 08 1920 JMP .1 NONE OF THE ABOVE
1930 *-----
1940 HANDLE.RIGHT.ARROW
1950 * MOVE UP ONE ELEMENT
0856- 20 E0 0A 1960 JSR CHECK.FOR.END.OF.ARRAY
0859- B0 0B 1970 BCS .2 DO NOTHING IF ALREADY AT
085B- 2C 37 0B 1980 BIT MOVING.FLAG SKIP SWAP IF END
085E- 10 03 1990 BPL .1 NOT MOVING
0860- 20 F3 0A 2000 JSR MOVE.ELEMENT.UP
0863- EE 36 0B 2010 INC ACTIVE.ELEMENT FOLLOW IT UP
0866- 4C 28 08 2020 .1 JMP DISPLAY.AND.READ.KEY
2030 .2
2040 *-----
2050 HANDLE.LEFT.ARROW
2060 * MOVE DOWN ONE ELEMENT
0869- 20 EA 0A 2060 JSR CHECK.FOR.BEGINNING.OF.ARRAY
086C- B0 0B 2070 BCS .2 IF AT BEGINNING, DO NOTHING
086E- 2C 37 0B 2080 BIT MOVING.FLAG IF NOT MOVING,
0871- 10 03 2090 BPL .1 SKIP SWAP
0873- 20 0D 0B 2100 JSR MOVE.ELEMENT.DOWN
0876- CE 36 0B 2110 .1 DEC ACTIVE.ELEMENT
0879- 4C 28 08 2120 .2 JMP DISPLAY.AND.READ.KEY
2130 *-----
2140 HANDLE.B
2150 * MOVE CURSOR TO BEGINNING OF ARRAY
087C- 20 EA 0A 2160 .1 JSR CHECK.FOR.BEGINNING.OF.ARRAY
087F- B0 0D 2170 BCS .3 DO NOTHING IF AT BEGINNING
0881- 2C 37 0B 2180 BIT MOVING.FLAG
0884- 10 03 2190 BPL .2
0886- 20 0D 0B 2200 JSR MOVE.ELEMENT.DOWN
0889- CE 36 0B 2210 .2 DEC ACTIVE.ELEMENT
088C- 10 EE 2220 BPL .1
088E- 4C 28 08 2230 .3 JMP DISPLAY.AND.READ.KEY
2240 *-----
2250 HANDLE.E
2260 * MOVE CURSOR TO END OF ARRAY
0891- 20 E0 0A 2270 .1 JSR CHECK.FOR.END.OF.ARRAY
0894- B0 0D 2280 BCS .3
0896- 2C 37 0B 2290 BIT MOVING.FLAG
0899- 10 03 2300 BPL .2
089B- 20 F3 0A 2310 JSR MOVE.ELEMENT.UP
089E- EE 36 0B 2320 .2 INC ACTIVE.ELEMENT
08A1- 10 EE 2330 BPL .1 ...ALWAYS
08A3- 4C 28 08 2340 .3 JMP DISPLAY.AND.READ.KEY
2350 *-----
2360 HANDLE.W
2370 * WRITE CATALOG TO DISK
08A6- 20 40 09 2380 JSR WRITE.CATALOG
08A9- 4C 28 08 2390 JMP DISPLAY.AND.READ.KEY
2400 *-----
2410 HANDLE.RETURN
2420 * TOGGLE MOVING FLAG
2430 * =FF IF MOVING
2440 * =0 IF NOT
08AC- AD 37 0B 2450 LDA MOVING.FLAG
08AF- 49 FF 2460 EOR #FF
08B1- 8D 37 0B 2470 STA MOVING.FLAG
08B4- 4C 28 08 2480 JMP DISPLAY.AND.READ.KEY
2490 *-----
2500 HANDLE.ESC
2510 * EXIT PROGRAM
08B7- 4C D0 03 2520 JMP DOS.RESTART
2530 *-----
2540 HANDLE.R
2550 * READ NEW CATALOG
08BA- 4C 03 08 2560 JMP SETUP RESTART PROGRAM
2570 *-----
2580 READ.CATALOG
08BD- 20 00 09 2590 JSR READ.VTOC
08C0- 20 8A 09 2600 JSR POINT.TO.FIRST.CATALOG.SECTOR
08C3- 20 1C 09 2610 .1 JSR READ.CATALOG.SECTOR
08C6- B0 24 2620 BCS .4 .CS. IF END OF CHAIN

```



MACHINE LANGUAGE SPEED WHERE IT COUNTS... IN YOUR PROGRAM!

Some routines on this disk are:

- Binary file info
- Delete array
- Disassemble memory
- Dump variables
- Find substring
- Get 2-byte values
- Gosub to variable
- Goto to variable
- Hex memory dump
- Input anything
- Move memory
- Multiple poke decimal
- Multiple poke hex
- Print w/o word break
- Restore special data
- Speed up Applesoft
- Speed restore
- Store 2-byte values
- Swap variables

For the first time, Amper-Magic makes it easy for people who don't know machine language to use its power! Now you can attach slick, finished machine language routines to your Applesoft programs in seconds! And interface them by name, not by address!

You simply give each routine a name of your choice, perform the append procedure once at about 15 seconds per routine, and the machine language becomes a permanent part of your BASIC program. (Of course, you can remove it if you want to.)

Up to 255 relocatable machine language routines can be attached to a BASIC program and then called by name. We supply some 20 routines on this disk. More can be entered from magazines. And more library disks are in the works.

These routines and more can be attached and accessed easily. For example, to allow the typing of commas and colons in a response (not normally allowed in Applesoft), you just attach the Input Anything routine and put this line in your program:

xxx PRINT "PLEASE ENTER THE DATE."; : & INPUT.DATES

&-MAGIC makes it Easy to be Fast & Flexible!

PRICE: \$75

**Anthro - Digital Software
P.O. Box 1385
Pittsfield, MA 01202**

The People - Computers Connection

&-Magic and Amper-Magic are trademarks of Anthro-Digital, Inc.
Applesoft is a trademark of Apple Computer, Inc.

```

2630
2640 * MOVE CATALOG SECTOR INTO ARRAY
2650 * X STEPS THROUGH BUFFER, $B-$FF
2660 * Y STEPS THROUGH ENTRY, 0-$23
08C8- A2 0B 2670 LDX $B
08CA- BD 38 0C 2680 .2 LDA CATALOG.BUFFER,X
08CD- F0 1D 2690 BEQ .4
08CF- EE 36 0B 2700 INC ACTIVE.ELEMENT NO, WE HAVE
08D2- EE 35 0B 2710 INC NUMBER.OF.ELEMENTS A NEW ENTRY
08D5- AD 36 0B 2720 LDA ACTIVE.ELEMENT
08D8- 20 C8 0A 2730 JSR POINT.TO.A SET POINTER
08DB- A0 00 2740 LDY #ZERO
08DD- BD 38 0C 2750 .3 LDA CATALOG.BUFFER,X
08E0- 91 00 2760 STA (POINTER),Y
08E2- E8 2770 INX
08E3- F0 DE 2780 BEQ .1 END OF BUFFER?
2790 * IF SO, READ NEW SECTOR
08E5- C8 2800 INY
08E6- C0 23 2810 CPY #ENTRY.LENGTH END OF ENTRY?
08E8- 90 F3 2820 BCC .3 NO, KEEP GOING
08EA- B0 DE 2830 BCS .2 YES, GET NEXT ONE
2840
08EC- AD 36 0B 2850 .4 LDA ACTIVE.ELEMENT
08EF- 18 2860 CLC GO ONE PAST
08F0- 69 01 2870 ADC #1 LAST ELEMENT
08F2- 0A 2880 ASL AND STORE
08F3- A8 2890 TAY TWO ZEROES
08F4- A9 00 2900 LDA #ZERO
08F6- 99 38 0D 2910 STA ARRAY.TABLE,Y
08F9- 99 39 0D 2920 STA ARRAY.TABLE+1,Y
08FC- 8D 36 0B 2930 STA ACTIVE.ELEMENT
08FF- 60 2940 RTS
2950 *-----
2960 READ.VTOC
0900- A9 00 2970 LDA #ZERO
0902- 8D 31 0B 2980 STA SECTOR
0905- A9 11 2990 LDA #$11
0907- 8D 30 0B 3000 STA TRACK
090A- A9 38 3010 LDA #VTOC.BUFFER
090C- 8D 32 0B 3020 STA BUFFER
090F- A9 0B 3030 LDA /VTOC.BUFFER
0911- 8D 33 0B 3040 STA BUFFER+1
0914- A9 01 3050 LDA #READ
0916- 8D 34 0B 3060 STA COMMAND
0919- 4C 3F 0A 3070 JMP RWTS.CALLER
3080 *-----
3090 READ.CATALOG.SECTOR
091C- AD 39 0C 3100 LDA CATALOG.BUFFER+1 GET NEXT TRACK
091F- F0 1D 3110 BEQ .1 END OF CATALOG CHAIN?
0921- 8D 30 0B 3120 STA TRACK
0924- AD 3A 0C 3130 LDA CATALOG.BUFFER+2 GET NEXT SECTOR
0927- 8D 31 0B 3140 STA SECTOR
092A- A9 38 3150 LDA #CATALOG.BUFFER
092C- 8D 32 0B 3160 STA BUFFER
092F- A9 0C 3170 LDA /CATALOG.BUFFER
0931- 8D 33 0B 3180 STA BUFFER+1
0934- A9 01 3190 LDA #READ
0936- 8D 34 0B 3200 STA COMMAND
0939- 20 3F 0A 3210 JSR RWTS.CALLER
093C- 18 3220 CLC
093D- 60 3230 RTS
3240
3250 * SET CARRY TO SHOW END-OF-CHAIN
093E- 38 3260 .1 SEC
093F- 60 3270 RTS
3280 *-----
3290 WRITE.CATALOG
0940- A9 FF 3300 LDA #$FF
0942- 8D 36 0B 3310 STA ACTIVE.ELEMENT
0945- 20 8A 09 3320 JSR POINT.TO.FIRST.CATALOG.SECTOR
0948- 20 1C 09 3330 .1 JSR READ.CATALOG.SECTOR
094B- A2 0B 3340 LDX $B
094D- EE 36 0B 3350 .2 INC ACTIVE.ELEMENT
0950- AD 36 0B 3360 LDA ACTIVE.ELEMENT
0953- 20 C8 0A 3370 JSR POINT.TO.A
0956- B0 17 3380 BCS .5 .CS. IF AT END OF TABLE
0958- A0 00 3390 LDY #ZERO
095A- B1 00 3400 .3 LDA (POINTER),Y
095C- 9D 38 0C 3410 STA CATALOG.BUFFER,X
095F- E8 3420 INX
0960- F0 07 3430 BEQ .4 END OF BUFFER?

```

```

0962- C8      3440      INY
0963- C0 23    3450      CPY #ENTRY.LENGTH END OF ENTRY?
0965- 90 F3    3460      BCC .3 NO, KEEP GOING
0967- B0 E4    3470      BCS .2 YES, GET NEXT ONE
              3480
0969- 20 82 09 3490      JSR WRITE.CATALOG.SECTOR
096C- 4C 48 09 3500      JMP .1 AND READ THE NEXT SECTOR
              3510
              * FILL THE REST OF THE BUFFER WITH ZEROES
096F- A9 00    3520      LDA #ZERO
0971- 9D 38 0C 3530      STA CATALOG.BUFFER,X
0974- E8      3540      INX
0975- D0 FA    3550      BNE .6
              3560
              3570
0977- 20 82 09 3580      JSR WRITE.CATALOG.SECTOR
097A- A9 00    3590      LDA #ZERO
097C- 8D 36 0B 3600      STA ACTIVE.ELEMENT
097F- 4C 28 08 3610      JMP DISPLAY.AND.READ.KEY
              3620
              *-----*
0982- A9 02    3630      WRITE.CATALOG.SECTOR
0984- 8D 34 0B 3640      LDA #WRITE WRITE THE SECTOR
0987- 4C 3F 0A 3650      STA COMMAND BACK JUST WHERE
              3660      JMP RWTS.CALLER IT CAME FROM
              3670
              *-----*
098A- AD 39 0B 3680      POINT.TO.FIRST.CATALOG.SECTOR
098D- 8D 39 0C 3690      * GET THE FIRST TRACK AND SECTOR FROM THE VTOC
0990- AD 3A 0B 3700      LDA VTOC.BUFFER+1
0993- 8D 3A 0C 3710      STA CATALOG.BUFFER+1
0996- 60      3720      LDA VTOC.BUFFER+2
              3730      STA CATALOG.BUFFER+2
              3740      RTS
              3750
              *-----*
0997- A9 00    3760      DISPLAY.ARRAY
0999- 85 25    3770      LDA #ZERO START AT
099B- 20 22 FC 3780      STA MON.CV TOP OF
099E- AD 36 0B 3790      JSR MON.VTAB SCREEN
              3800      LDA ACTIVE.ELEMENT
              3810      SEC
09A2- E9 0B    3820      SBC #LINE.COUNT/2
09A4- 10 02    3830      BPL .1 IF RESULT IS +, USE IT
09A6- A9 00    3840      LDA #ZERO OTHERWISE, USE ZERO
              3850      TAX X KEEPS TRACK OF
09A8- AA      3860      TXA WHERE WE ARE
09AA- CD 36 0B 3870      CMP ACTIVE.ELEMENT
09AD- D0 05    3880      BNE .3
09AF- 48      3890      PHA
09B0- 20 80 FE 3900      JSR MON.SETINV INVERT ACTIVE ELEMENT
09B3- 68      3910      PLA
09B4- 20 C8 0A 3920      JSR POINT.TO.A SET POINTER
09B7- B0 12    3930      BCS .5 .CS. IF AT END OF TABLE
09B9- 20 DA 09 3940      JSR INTERPRET.ENTRY WRITE A LINE
09BC- A9 8D    3950      LDA #RETURN
09BE- 20 F0 FD 3960      JSR MON.COUT1
09C1- 20 84 FE 3970      JSR MON.SETNORM RESTORE NORMAL
09C4- E8      3980      INX
09C5- A5 25    3990      LDA MON.CV
09C7- C9 16    4000      CMP #LINE.COUNT END OF SCREEN?
09C9- 90 DE    4010      BCC .2 IF NOT, DO ANOTHER LINE
09CB- 20 42 FC 4020      JSR MON.CLREOP CLEAR TO END OF PAGE
09CE- 20 84 FE 4030      JSR MON.SETNORM JUST IN CASE
09D1- 2C 37 0B 4040      BIT MOVING.FLAG
09D4- 10 03    4050      BPL .6
09D6- 20 33 0A 4060      JSR SHOW.MOVING.FLAG IF MOVING
09D9- 60      4070      RTS
              4080
              *-----*
09DA- A0 00    4090      INTERPRET.ENTRY
09DC- B1 00    4100      LDY #ZERO
09DE- 10 04    4110      LDA (POINTER),Y DELETED?
09E0- A9 AD    4120      BPL .1 MINUS IF YES
09E2- 30 0C    4130      LDA #AD -
09E4- A0 02    4140      BMI .3 ...ALWAYS
09E6- B1 00    4150      LDY #2
09E8- 10 04    4160      LDA (POINTER),Y LOCKED?
09EA- A9 AA    4170      BPL .2 MINUS IF YES
09EC- 30 02    4180      LDA #AA *
09EE- A9 A0    4190      BMI .3 ...ALWAYS
09F0- 20 F0 FD 4200      LDA #SPACE NEITHER DELETED NOR LOCKED
09F0- 20 F0 FD 4210      JSR MON.COUT1

```

09F3-	A0	02	4220	LDY	#2	
09F5-	B1	00	4230	LDA	(POINTER),Y	GET FILE TYPE
09F7-	29	7F	4240	AND	#\$7F	MAKE POINTER
09F9-	A0	07	4250	LDY	#7	INTO DOS'S
09FB-	0A		4260	ASL		TYPE TABLE
09FC-	0A		4270	ASL		(ROUTINE BORROWED
09FD-	B0	03	4280	BCS	.5	FROM DOS, \$ADE8-ADF9)
09FF-	88		4290	DEY		
0A00-	D0	FA	4300	BNE	.4	
0A02-	B9	A7 B3	4310	LDA	DOS.TYPTABL,Y	
0A05-	20	F0 FD	4320	JSR	MON.COUT1	DISPLAY TYPE
0A08-	A9	A0	4330	LDA	#SPACE	
0A0A-	20	F0 FD	4340	JSR	MON.COUT1	
0A0D-	A0	21	4350	LDY	#\$21	
0A0F-	B1	00	4360	LDA	(POINTER),Y	SET UP FOR
0A11-	85	44	4370	STA	\$44	ROUTINE TO
0A13-	C8		4380	INY		DISPLAY FILE
0A14-	B1	00	4390	LDA	(POINTER),Y	SIZE
0A16-	85	45	4400	STA	\$45	
0A18-	20	42 AE	4410	JSR	DOS.SIZEOUT	DO IT
0A1B-	A9	A0	4420	LDA	#SPACE	
0A1D-	20	F0 FD	4430	JSR	MON.COUT1	
0A20-	A0	03	4440	LDY	#3	
0A22-	B1	00	4450	LDA	(POINTER),Y	GET A CHARACTER
0A24-	C9	A0	4451	CMP	#SPACE	CONTROL?
0A26-	B0	02	4452	BCS	.7	NO, GO ON
0A28-	29	7F	4453	AND	#\$7F	YES, MAKE IT INVERSE
0A2A-	20	F0 FD	4460	JSR	MON.COUT1	DISPLAY IT
0A2D-	C8		4470	INY		
0A2E-	C0	21	4480	CPY	#33	DONE WITH FILE NAME?
0A30-	90	F0	4490	BCC	.6	
0A32-	60		4500	RTS		

0A33-	A0	05	4510	SHOW.MOVING.FLAG		
0A35-	B9	27 0B	4520	LDY	#5	PUT INVERSE
0A38-	99	D0 07	4530	LDA	QMOVING,Y	"MOVING" AT
0A3B-	88		4540	STA	CORNER,Y	BOTTOM OF
0A3C-	10	F7	4550	DEY		SCREEN
0A3E-	60		4560	BPL	.1	
			4570	RTS		

0A3F-	AD	2D 0B	4580	RWTS.CALLER		
0A42-	8D	E9 B7	4590	LDA	SLOT	TRANSFER
0A45-	AD	2E 0B	4600	STA	IOB.SLOT	VALUES
0A48-	8D	EA B7	4610	LDA	DRIVE	INTO
0A4B-	AD	2F 0B	4620	STA	IOB.DRIVE	IOB
0A4E-	8D	EB B7	4630	LDA	VOLUME	
0A51-	AD	30 0B	4640	STA	IOB.VOLUME	
0A54-	8D	EC B7	4650	LDA	TRACK	
0A57-	AD	31 0B	4660	STA	IOB.TRACK	
0A5A-	8D	ED B7	4670	LDA	SECTOR	
0A5D-	AD	34 0B	4680	STA	IOB.SECTOR	
0A60-	8D	F4 B7	4690	LDA	COMMAND	
0A63-	AD	32 0B	4700	STA	IOB.COMMAND	
0A66-	8D	F0 B7	4710	LDA	BUFFER	
0A69-	AD	33 0B	4720	STA	IOB.BUFFER	
0A6C-	8D	F1 B7	4730	LDA	BUFFER+1	
0A6F-	A9	00	4740	STA	IOB.BUFFER+1	
0A71-	8D	F5 B7	4750	LDA	#\$00	
			4760	STA	IOB.ERROR	

0A74-	A0	E8	4770	LDY	#IOB	LOAD IOB
0A76-	A9	B7	4780	LDA	/IOB	ADDRESS
0A78-	20	D9 03	4790	JSR	DOS.RWTS	CALL RWTS
0A7B-	A9	00	4800	LDA	#\$00	
0A7D-	85	48	4810	STA	PREG	SOOTHE MONITOR
0A7F-	B0	01	4820	BCS	ERROR.HANDLER	
0A81-	60		4830	RTS		

0A82-	A9	87	4840	ERROR.HANDLER		
0A84-	20	F0 FD	4850	LDA	#\$87	BELL
0A87-	20	F0 FD	4860	JSR	MON.COUT1	RING
0A8A-	20	F0 FD	4870	JSR	MON.COUT1	ING
0A8D-	A9	17	4880	JSR	MON.COUT1	ING
0A8F-	85	25	4890	LDA	#23	
0A91-	20	22 FC	4900	STA	MON.CV	USE LINE BELOW DISPLAY
			4910	JSR	MON.VTAB	

AMPERWARE

THE ULTIMATE ENHANCEMENT TO APPLESOFT* BASIC

There are a number of Applesoft enhancement packages on the market which use the ampersand reserved word to gain access to assembly language routines from BASIC. None can match the ease of use, speed or powerful features of **Amperware**. Just BLOAD AMPERWARE and you have access to an extended instruction set which will make your programs smaller, faster, more professional and easier to write. No knowledge of assembly language is required and there are not POKes, PEEKs, CALLs or limitations on variables.

— FEATURES —

FLEXIBLE DATA ENTRY -- The &INPUT command allows rapid creation of video forms with line editing (insert/delete character, clear to end, etc.), upper/lower case and numeric error checking. Control codes handling is easy with the &ON CTRL GOTO statement.

RAPID DISK I/O -- Read or write numeric or string information to the disk is binary up to 20 times faster than in BASIC. Compact storage format saves 25-75% of disk space. &READ/&WRITE whole arrays or even lists of arrays with one statement.

FORMATTED OUTPUT -- &PRINT WITH will print in regular, exponential or accounting format (negative in parentheses). Complete control of: significant digits; decimal sign; dollar sign; comma placement; fill characters (blank, zero or asterisk). Handles text in the format and will place text into format fields with right/left justification or centering.

MANY OTHERS -- Sort 8-10 times faster (direct or indexed); delete arrays; rapid substring search; GOTO/GOSUB to computed line number; simple text window control; hardware independent cursor and screen control; etc.

ALL THIS FOR ONLY \$49.95

add \$1.50 shipping and handling

SCIENTIFIC SOFTWARE PRODUCTS, INC.

3171 Donald Avenue
Indianapolis, IN 46224

IN residents add 4% tax

VISA and MC accepted

* Applesoft is a trademark of Apple Computer Co., Inc.

```

0A94- A2 08 4960      LDX #8
0A96- 20 02 A7 4970      JSR DOS.PRNTERR      DISPLAY "I/O ERROR"
0A99- 4C D0 03 4980      JMP DOS.RESTART      EXIT PROGRAM
                                *-----*
                                BUILD.ARRAY.TABLE
0A9C- A9 38 5010      LDA #CATALOG.ARRAY      SET FIRST ENTRY
0A9E- 8D 38 0D 5020      STA ARRAY.TABLE      TO POINT TO
0AA1- A9 0E 5030      LDA /CATALOG.ARRAY      START OF
0AA3- 8D 39 0D 5040      STA ARRAY.TABLE+1      ARRAY
0AA6- A2 02 5050      LDX #2
0AA8- BD 36 0D 5060      LDA ARRAY.TABLE-2,X      MAKE EACH
0AAB- 18 5070      CLC                          SUCCESSIVE
0AAC- 69 23 5080      ADC #ENTRY.LENGTH      ENTRY #23
0AAE- 9D 38 0D 5090      STA ARRAY.TABLE,X      LARGER THAN
0AB1- BD 37 0D 5100      LDA ARRAY.TABLE-1,X      THE LAST
0AB4- 69 00 5110      ADC #ZERO
0AB6- 9D 39 0D 5120      STA ARRAY.TABLE+1,X
0AB9- E8 5130      INX
0ABA- E8 5140      INX
0ABB- E0 FE 5150      CPX #$FE      127 ENTRIES YET?
0ABD- D0 E9 5160      BNE .1
0ABF- A9 00 5170      LDA #ZERO      END TABLE
0AC1- 9D 38 0D 5180      STA ARRAY.TABLE,X      WITH TWO
0AC4- 9D 39 0D 5190      STA ARRAY.TABLE+1,X      ZEROES
0AC7- 60 5200      RTS
                                *-----*
                                POINT.TO.A
0AC8- 0A 5220      ASL                          MAKE (A) INTO INDEX
0AC9- A8 5230      TAY
0ACA- B9 38 0D 5240      LDA ARRAY.TABLE,Y      CHECK FOR TWO
0ACD- 19 39 0D 5250      ORA ARRAY.TABLE+1,Y      CONSECUTIVE
0ADO- F0 0C 5260      BEQ .1      ZERO BYTES
0AD2- B9 38 0D 5270      LDA ARRAY.TABLE,Y
0AD5- 85 00 5280      STA POINTER      PUT TABLE ENTRY
0AD7- B9 39 0D 5290      LDA ARRAY.TABLE+1,Y      INTO POINTER
0ADA- 85 01 5300      STA POINTER+1
0ADC- 18 5310      CLC
0ADD- 60 5320      RTS
                                *-----*
0ADE- 38 5330      .1      SEC      END OF TABLE
0ADF- 60 5340      RTS
                                *-----*
0AE0- AD 36 0B 5350      CHECK.FOR.END.OF.ARRAY
0AE3- 18 5360      * RETURNS CARRY SET IF AT END
0AE4- 69 01 5370      * " " CLEAR IF NOT
0AE6- CD 35 0B 5380      LDA ACTIVE.ELEMENT
0AE9- 60 5390      CLC
0AE0- AD 36 0B 5400      ADC #1
0AE3- 18 5410      CMP NUMBER.OF.ELEMENTS
0AE6- CD 35 0B 5420      RTS
                                *-----*
0AEA- AD 36 0B 5430      CHECK.FOR.BEGINNING.OF.ARRAY
0AED- D0 02 5440      LDA ACTIVE.ELEMENT
0AEF- 38 5450      BNE .1
0AFO- 60 5460      SEC      ACTIVE = 0, WE ARE AT BEGINNING
0AF1- 18 5470      CLC
0AF2- 60 5480      RTS      NONZERO, WE'RE OKAY
                                *-----*
0AF3- AD 36 0B 5490      MOVE.ELEMENT.UP
0AF6- 0A 5500      LDA ACTIVE.ELEMENT
0AF7- AA 5510      ASL      MAKE INDEX INTO TABLE
0AF8- A0 02 5520      TAX
0AFA- BD 38 0D 5530      LDY #2      DO THIS TWICE, FIRST LO, THEN
0AFD- 48 38 0D 5540      LDA ARRAY.TABLE,X      HI
0AFE- BD 3A 0D 5550      PHA
0B01- 9D 38 0D 5560      LDA ARRAY.TABLE+2,X
0B04- 68 38 0D 5570      STA ARRAY.TABLE,X
0B05- 9D 3A 0D 5580      PLA
0B08- E8 5590      STA ARRAY.TABLE+2,X
0B09- 88 5600      INX      NOW DO HIGH BYTES
0B0A- D0 EE 5610      DEY
0B0C- 60 5620      BNE .1      DONE?
                                *-----*
0B0C- 60 5630      RTS
0B0C- 60 5640      RTS
0B0C- 60 5650      RTS
0B0C- 60 5660      RTS
0B0C- 60 5670      RTS
0B0C- 60 5680      RTS
0B0C- 60 5690      RTS
0B0C- 60 5700      RTS
0B0C- 60 5710      RTS

```

```

5720 MOVE.ELEMENT.DOWN
OB0D- AD 36 OB 5730 LDA ACTIVE.ELEMENT
OB10- OA 5740 ASL
OB11- AA 5750 TAX
OB12- A0 02 5760 LDY #2
OB14- BD 38 OD 5770 .1 LDA ARRAY.TABLE,X
OB17- 48 5780 PHA
OB18- BD 36 OD 5790 LDA ARRAY.TABLE-2,X
OB1E- 9D 38 OD 5800 STA ARRAY.TABLE,X
OB1E- 68 5810 PLA
OB1F- 9D 36 OD 5820 STA ARRAY.TABLE-2,X
OB22- E8 5830 INX
OB23- 88 5840 DEY
OB24- D0 EE 5850 BNE .1
OB26- 60 5860 RTS
5870 -----
5880 QMOVING
5890 # INVERSE "MOVING"

OB27- OD OF 16 5900 .HS 0D0F16090E07
OB2A- 09 OE 07 5910 -----
OB2D- 5920 SLOT .BS 1 (USUALLY 6)
OB2E- 5930 DRIVE .BS 1 (USUALLY 1)
OB2F- 5940 VOLUME .BS 1 (0 = ANY)
OB30- 5950 TRACK .BS 1 (USUALLY $11)
OB31- 5960 SECTOR .BS 1 (0 TO F)
OB32- 5970 BUFFER .BS 2 (VARIES)
OB34- 5980 COMMAND .BS 1 (1 OR 2)
5990 -----
OB35- 6000 NUMBER.OF.ELEMENTS .BS 1 (1 TO N)
OB36- 6010 ACTIVE.ELEMENT .BS 1 (0 TO N-1)
OB37- 6020 MOVING.FLAG .BS 1 (0 OR FF)
6030 -----
6040 END.OF.PROGRAM
6050 -----
OB38- 6060 VTOC.BUFFER .EQ END.OF.PROGRAM
OC38- 6070 CATALOG.BUFFER .EQ END.OF.PROGRAM+$100
OD38- 6080 ARRAY.TABLE .EQ END.OF.PROGRAM+$200
OE38- 6090 CATALOG.ARRAY .EQ END.OF.PROGRAM+$300

```

write now

Southwestern Data Systems, an industry pioneer in innovative software for the Apple II, is always looking for authors. There are no limitations on the size or type of software you can submit — utilities, communication, business, education, or games — the only requirement is that it must meet the quality standards which typify all SDS products. When you join the SDS team, you get the benefits of a professional support staff experienced in providing all you need to get your program to market. Here are some of the ways we help you:

- TECHNICAL PROGRAMMING ASSISTANCE
- UNIQUE COPY PROTECTION W/LIMITED BACKUPS
- SUCCESSFUL MARKETING STRATEGIES
- ASSISTANCE IN WRITING THE MANUAL
- PROFESSIONAL PRODUCT ARTWORK
- QUALITY ADVERTISING
- SUPERIOR PACKAGING
- NATIONAL DISTRIBUTION
- HIGHEST ROYALTIES PAID MONTHLY
- CUSTOMER SERVICE SUPPORT

This is the opportunity you have been waiting for, a chance to market your program with the finest publisher in the software industry. Let Southwestern Data Systems' reputation and proven track record for success go to work for you. If you think you have what we want — a unique and distinctive software package — please call or write us today!

SDS southwestern data systems

P.O. BOX 582 SANTEE, CA 92071 (714) 562-3670

So You Never Need Macros!.....Bob Sander-Cederlof

I have said it many times myself, "I don't need macros!" But now that I have them, I seem to find more and more uses for them. Not the traditional uses, to generate common sequences of opcodes. I am using them to build tables of data, rather than typing in line after line of very similar stuff.

I have been working some more on the Prime Number Generator program. You may remember the series: first the articles in BYTE Magazine, then my faster version in an early Apple Assembly Line, then Charles Putney's version at double my speed. Now Tony Brightwell has cut Charlie's time nearly in half. (His program will probably appear next month.) Anyway, I have done some more investigation.

One approach required a precomputed table of the squares of the odd numbers from 1 to 127. An easy way to enter this table might be:

```
.DA 1*1,3*3,5*5,7*7,9*9
.DA 11*11,13*13,15*15,17*17
et cetera
```

I had type about that much when I said, "There has to be an easier way." I made up the following macro definition:

```
.MA SQ
:0 .EQ ]1
:1 .EQ ]1+2
:2 .EQ ]1+4
:3 .EQ ]1+6
:4 .EQ ]1+8
:5 .EQ ]1+10
:6 .EQ ]1+12
:7 .EQ ]1+14
.DA :0*:0,:1*:1,:2*:2,:3*:3
.DA :4*:4,:5*:5,:6*:6,:7*:7
.DO ]2<8
>SQ ]1+16,]2+1
.FIN
.EM
```

Then the single line of code

```
2200 >SQ 1,1
```

generated all 64 squares for me.

How does it work? Good question.... The eight .EQ lines create 8 private labels with the values of 8 consecutive odd numbers starting with whatever the first parameter from the call line happens to be. Line 2200 has the first parameter "1", so the private labels will have values of 1, 3, 5, 7, 9, 11, 13, and 15 respectively. The two .DA lines generate the squares of these 8 values.

The next three lines are the tricky part. If the second parameter has a value less than 8 then the line between .DO and .FIN is assembled. It is a nested call on the SQ macro. Only this time the first parameter is 16 greater than it was, and the second parameter is one greater. After going through this nesting process 7 times, we will have generated 8 sets of 8 values each. When the second parameter has worked its way up to 8, the nested calls will exit in turn, and the table is finished.

If you have the macro expansion listing option on during assembly, the expanded form takes 2 1/2 pages.

DISASM (Version 2.2)

\$30.00

Use DISASM, the intelligent disassembler, to convert 6502 machine code into meaningful, symbolic source. It creates a text file which is directly compatible with DOS ToolKit, LISA and S-C (both 4.0 & Macro) Assemblers. Use DISASM to customize existing machine language programs to your own needs or just to see how they work. DISASM handles multiple data tables, invalid op codes and displaced object code (the program being disassembled doesn't have to reside in the memory space in which it executes). DISASM lets you even substitute MEANINGFUL labels of your own choice (100 commonly used Monitor & Pg Zero names included in Source form to get you rolling). The address-based cross reference table option results in either a selective or complete cross reference (to either screen or printer). Page Zero and External references are listed separately in numeric order. The cross reference table provides as much insight into the inner workings of machine language programs as the disassembly itself. DISASM has proven to be an invaluable aid for both the novice and expert alike.

Utilities For Your S-C Assembler (4.0)

SC.GSR: A Global Search and Replace Eliminates Tedious Manual Renaming Of Labels.....\$20.00
SC.XREF: A Linenumber-Based Global Cross Reference Table For Complete Source Documentation.....\$20.00
SC.TAB: Tabulates Source Files Into Neat, Readable Form. Encourages Fast, Free-Format Entry....\$15.00
SC UTILITY PAK: Includes All Three Utilities Described Above (You Save \$10.00).....\$45.00

All of the above programs are written entriely in machine language and are provided
on a standard 3.3 DOS formatted diskette.

Avoid A \$3.00 Shipping/Handling Charge By Mailing Full Payment With Order

R A K - W A R E
41 Ralph Road
West Orange NJ 07052

**** SAY YOU SAW IT IN 'APPLE ASSEMBLY LINE'! ****



S&H Software is Speeding up The Apple!

**The DOS Enhancer (TDE)
works up to 500% faster than
standard Apple DOS 3.3...\$69.95**

TIRED OF WAITING... for your programs to load or save? Then S&H's TDE, licensed by Apple, is the answer.

TDE program updates standard Apple DOS 3.3 disks... or creates copyable TDE disks... with TDE's QuickDOS and Quick load features.

TDE's QuickDOS runs and saves BASIC and binary programs up to 5 times faster* and is completely compatible with standard Apple DOS 3.3 programs.

TDE "Quick loads" the RAM card with FPHASIC/I... QuickDOS or user program in 1.7 seconds at startup.

TDE "package" includes utility disk, training/support disk, step-by-step instruction manual, S&H's Supercat/menu, and multidrive "Quick-copy" program.

TDE system requirements: 48K Apple II or II+ ROM/RAM card, DOS 3.3 and one or more disk drives.

Here's what the critics say:

● John Mitchener of PEELINGS II "The speed increase with TDE is awesome and is probably worth the price of the program alone... without all the other features... AA rating

● Val Golding, Editor of Call APPLE II (TDE) stands as a shining example of how utility and application programs take into account every possible system configuration.

● Clark Conleton of The Apple Orchard "The Quick load capabilities will make this package attractive to anyone who spends a lot of time at the keyboard."

● Chuck Carpenter of INFOWORD "Results in a disk that loads... in any Apple system."

**Amper-Sort/Merge (A-S/M II)
works up to 1000% faster than even
VisiCorp's VisiFile program...\$69.95**

A-S/M II is the fastest "file clerk" you've ever met. Of all the sort utilities developed to manage Apple II data files, none does the job nearly so fast as A-S/M II.

A-S/M II's new features include: S&H's superfast VisiFile index sort (callable from within VisiFile for effortless use), an equally fast S&H random access file index sort, and parameter file editing.

A-S/M II can sort/merge from one to five unsorted files into a single file of up to 125K in size per disk.

A-S/M II's "package" includes utility disk, training disk, step by step instruction manual, and S&H's new Supercat/menu.

A-S/M II's system requirements: 48K Apple II with ROM or RAM card, 48K Apple II+ with DOS 3.3 and Disk II.

**S&H
Software**

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203
817/382-6353

DIS-ASSEMBLER

DSA-DS dis-assembles Apple machine language programs into forms compatible with LISA, S-C ASSEMBLER (3.2 or 4.0), Apple's TOOL-KIT ASSEMBLER and others. DSA-DS dis-assembles instructions or data. Labels are generated for referenced locations within the machine language program.

\$25, Disk, Applesoft (32K, ROM or Language card)

OTHER PRODUCTS

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your **BASIC** programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.

\$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Use advanced logic constructs such as **IF...ELSE...**, **CASE**, **SELECT**, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of **PASCAL**.

\$35. Disk, Applesoft (48K, ROM or Language Card).

FORM-DS is a complete system for the definition of input and output forms. **FORM-DS** supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.

\$25 Disk, Applesoft (32K, ROM or Language Card).

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's **CLEAR** gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.

\$25 Disk, Applesoft.

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, **SPEED-DS** includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.

\$15 Disk, Applesoft (32K, ROM or Language Card).

(Add \$4.00 for Foreign Mail)

*Apple II is a registered trademark of the Apple Computer Co.

I had the source code for FIG-FORTH on the Apple, entered by some members of the Dallas Apple Corps. For some reason they decided to use the DOS ToolKit Assembler when they typed in all those lines. Naturally, I had a strong desire to convert the source files to the format of my S-C Macro Assembler.

The first step, and one of the easiest in this case, is to figure out how to read the ToolKit source files into S-C. ToolKit source files are standard DOS text files (type "T"). There are no line numbers. S-C allows such files to be read in by typing the following commands:

```
NEW
AUTO
<<<<<EXEC filename    (where "<" stands for backspace)
<<<<<MANUAL
```

"NEW" makes sure there are no lingering program lines from a previous load. "AUTO" starts generating automatic line numbers. The first line number generated will be 1000. Five backspaces will back up the cursor to the beginning of the input line, so the EXEC command can be typed. As the file is EXECing, each line will be read in with a prefixed line number. After the whole file has been read, five backspaces allow you to type the "MANUAL" command, thereby turning off the AUTO mode.

At this point you can LIST the program in memory and see what a ToolKit file looks like when you are using the S-C editor. You could use the EDIT and REPLACE commands to make all the necessary changes, and SAVE the converted program on a new file.

I was able to automate much of the conversion process, using an EXEC file of REPLACE commands. Several of you readers, including Graeme Scott of DFX fame, have sent me similar EXEC files for converting LISA source code.

Before I lay out the whole file, let's look at a simple case. The people who typed in the ToolKit source decided to separate individual sections of code with "SKP 1" lines. This causes a blank line on the assembly listing. S-C does not have an equivalent directive, but then again I personally don't like blank lines on my listings. (They always make me think my printer is broken!) Anyway, the command REP / SKP 1/*A replaces all of the skips with empty comment lines. If you don't even want to see the asterisk on the line, use REP / SKP 1/ /A.

Notice that there is one space before "SKP" in the command above. ToolKit uses space as a tab character, and so the source file does not have nice neat columns for each field. If you list it with a regular text editor, the opcode field winds around like a snake; it always starting one space after the label, or in column 2 if there is no label. S-C uses control-I for a tab character, because control-I is the ASCII tab character. More on this later.

There were also a number of SKP 2" lines. I decided to turn these into "*-----" lines, to indicate a greater separation than a mere empty comment line would.

ToolKit uses the semi-colon in column 1 to indicate a comment line; S-C uses an asterisk. ToolKit also uses a semi-colon to begin a comment field on a source line; S-C does not require any such character. The following two replace commands will make the necessary changes:

```
REP / ;/ /A
REP / ;/* /A
```

The commands have to be in that order, or else you end up with an asterisk starting comment fields when they aren't necessary. Two lines had ";" in as ASCII literal constant. I had to hand-re-correct them later.

The most important changes are the directives. The files I was converting needed the following changes:

```
REP / EQU / .EQ /A
REP / DW / .DA /A
REP / ORG / .OR /A
REP / DS / .BS /A
REP / DCI / .AT /A
REP / DFB / .DA #/
REP / ASC / .AS /
```

Immediate address mode also presented a problem. ToolKit uses the form "LDA #<SSS" to indicate the high byte, and "LDA #>SSS" to indicate the low byte. S-C uses "LDA /SSS" for the high byte, and "LDA #SSS" for the low byte. I fixed them with:

```
REP " <#" /"A
REP " >#" #"A
```

Now about those snaky columns.... I wanted to somehow put a tab before each opcode field, and before each comment field. I thought, "Why not just use the replace command to put in a control-I?":

```
REP / EQU / ^I.EQ /A      (where ^I means I typed control-I)
et cetera
```

My first problem was that typing control-I when entering the REPLACE command made a tab. I overcame that by typing the sequence "control-O control-I". Control-O makes the next character become part of the input line regardless of its normal meaning. That worked, but....

My second problem was that getting a control-I into the source program did not make it a tab. Somehow the control-I had to be "executed". So I wrote the converted program on a text file, this time with line numbers, and then EXECed it back in.

```
TEXT# filename
EXEC filename
```

the most controversial computer magazine

"When some Apple enthusiasts heard about the boycott (of bit-copy ads), they concluded that it was nothing but censorship and another example of the magazines ignoring the average Apple user to placate their advertisers. So they started their own publication, **HARDCORE Computing**" *ESQUIRE*, Jan. 1982

"HARDCORE Computing warns pirates about the latest technology that companies are using against them." *TIME*, Feb. 8, 1982

what is hardcore computing ? it's a "HOW TO" guide for users

**about
DOS**

How To Write Your Own Data Base
Using text files and EXEC
CALLING DOS routines from BASIC
source listings of special DOS routines

applesoft tricks 'n treats
ampersand (&) utilities
subroutine master library

**PEEKs
and
POKEs**

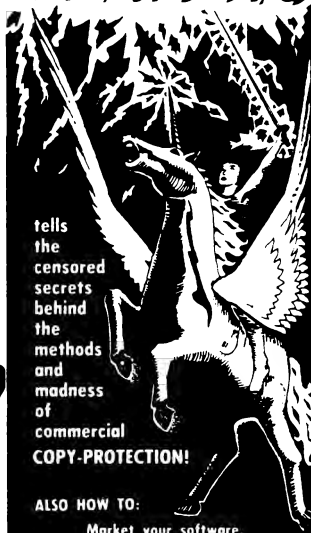
plus...

Lots of complete program listings from the
HARDCORE Program Library of copyable software

upcoming...

no. 4 All About Graphics!
no. 5 Program Utilities.
no. 6 Games?

hardcore



tells
the
censored
secrets
behind
the
methods
and
madness
of
commercial
COPY-PROTECTION!

ALSO HOW TO:

Market your software.
Copy-protect your disks.
Normalize altered D.O.S.
Use bit-copiers to make back-ups.
PLUS Game, Utility, Business,
Educational program listings.

For serious
Apple computerists!

HARDCORE computing

Dept. AL SUBSCRIPTION: \$20.00 (U.S.A.)

P.O. Box 44549 \$32 Mexico \$29 Canada \$42 others

Tacoma, WA 98444

Sample copy \$5.00 U.S.A. \$7.00 elsewhere

HardCore Computing is a quarterly magazine.

That "executed" the control-I's, and I had tabs. But....

My third problem was that I wanted to save all the REPLACE commands as an EXEC file, so that I did not have to manually retype them for every file to be converted. When I EXECed the REPLACE command file, the control-I's were executed immediately! I had to change my replace commands to include both a control-O and a control-I, so that the control-I in the REPLACE command would be read in from the EXEC file but not executed until it was later EXECed from the temporary source text file.

Still with me? If not, keep reading anyway, because I will show you what I mean.

Using S-C, I entered the following "program":

```
1000 "TOOLKIT CONVERTER
1010 "
1020 "
1030 "COMMENTS
1040 REP/ SKP 1/* /A
1050 REP/ SKP 2/*O[L/A
1060 REP/ ;/*O I/A
1070 REP/ ;/* /A
1080 "DIRECTIVES
1090 REP/ EQU /*Q^I.EQ /A
1100 REP/ DW /*Q^I.DA /A
1110 REP/ ORG /*Q^I.OR /A
1120 REP/ DS /*Q^I.BS /A
1130 REP/ DCI /*Q^I.AT /A
1140 REP/ ASC /*Q^I.AS /A
1150 REP/ DFB /*Q^I.DA #/A
1160 REP/ CHN /**** CHN /A
1170 "OPCODE TABS
1180 REP/ ADC /*Q^IADC /A
1190 REP/ AND /*Q^IAND /A
1200 REP/ ASL /*Q^IASL /A
1210 REP/ BIT /*Q^IBIT /A
1220 REP/ CMP /*Q^ICMP /A
1230 REP/ CPX /*Q^ICPX /A
1240 REP/ CPY /*Q^ICPY /A
1250 REP/ DEC /*Q^IDEC /A
1260 REP/ EOR /*Q^IEOR /A
1270 REP/ INC /*Q^IINC /A
1280 REP/ LDA /*Q^ILDA /A
1290 REP/ LDX /*Q^ILDX /A
1300 REP/ LDY /*Q^ILDY /A
1310 REP/ LSR /*Q^ILSR /A
1320 REP/ ORA /*Q^IORA /A
1330 REP/ ROL /*Q^IROL /A
1340 REP/ ROR /*Q^IROR /A
1350 REP/ SBC /*Q^ISBC /A
1360 REP/ STA /*Q^ISTA /A
1370 REP/ STX /*Q^ISTX /A
1380 REP/ STY /*Q^ISTY /A
1390 REP/ BPL /*Q^IBPL /A
1400 REP/ BMI /*Q^IBMI /A
1410 REP/ BEQ /*Q^IBEQ /A
1420 REP/ BNE /*Q^IBNE /A
1430 REP/ BVS /*Q^IBVS /A
1440 REP/ BVC /*Q^IBVC /A
1450 REP/ BCC /*Q^IBCC /A
1460 REP/ BCS /*Q^IBCS /A
1470 REP/ JMP /*Q^IJMP /A
1480 REP/ JSR /*Q^IJSR /A
1490 REP/ BRK /*Q^IBRK /A
1500 REP/ CLC /*Q^ICLC /A
1510 REP/ CLD /*Q^ICLD /A
1520 REP/ CLV /*Q^ICLV /A
1530 REP/ DEX /*Q^IDEX /A
1540 REP/ DEY /*Q^IDEY /A
1550 REP/ INX /*Q^IINX /A
1560 REP/ INY /*Q^IINY /A
1570 REP/ NOP /*Q^INOP /A
1580 REP/ PHA /*Q^IPHA /A
1590 REP/ PLA /*Q^IPLA /A
1600 REP/ PLP /*Q^IPLP /A
1610 REP/ RTS /*Q^IRTS /A
1620 REP/ SEC /*Q^ISEC /A
1630 REP/ TAX /*Q^ITAX /A
1640 REP/ TAY /*Q^ITAY /A
1650 REP/ TSX /*Q^ITSX /A
1660 REP/ TXA /*Q^ITXA /A
1670 REP/ TXS /*Q^ITXS /A
1680 REP/ TYA /*Q^ITYA /A
1690 "ADDRESS MODES
1700 REP" #>" #^A
1710 REP" #<" #^A
1720 "WRITE ON TEMP FILE
1730 TEXT#F
```


In the listing above, I have used "^O" to mean "control-O"; "^I" to mean "control-I"; and "^[" to mean "ESCAPE key". In order to get "control-O control-I" in a line, I had to type "control-O control-O control-O control-I".

Lines 1000-1030, 1080,1130, 1170,1690, and 1720 begin with a quotation mark. These are comment lines to the S-C input routine; they print on the screen when they are read from the EXEC file, but are otherwise ignored.

I saved the file as is using "SAVE TOOLKIT CONVERTER", in case I might want to modify it again. And I did, again and again and again. Then I wrote it on a text file without line numbers using "TEXT TKC".

Here is the sequence of steps I went through for each source file:

```
NEW
AUTO
1000 *      filename
<<<<<EXEC filename,D2      (where "<" means "backspace")
<<<<<MAN
EXEC TKC,D1
EXEC F
LIST 1000              (to see what filename to use)
SAVE filename
```

There are three EXEC commands above; the first reads on the ToolKit source file; the second executes all the REPLACE commands, and writes the resulting source on a temporary text file named "F"; the third reads in that temporary text file to "execute" the control-I tabs and the "ESC-L" lines.

After all the files were converted, I built a little assembly control file like this:

```
1000          .IN FILE1
1010          .IN FILE2
      et cetera
```

I also added a ".TF" directive after the ".OR" line, to put the assembled code on a DOS binary file.

The first assembly did not go smoothly, because of lines containing "ROR A". In the four shift instructions, ToolKit requires the symbol "A" to signify Accumulator mode. S-C uses a blank operand field to signify Accumulator mode, and thinks "ROR A" means to shift the memory location labeled "A".

Once I was able to assemble with no errors, I compared the object code produced with that produced by ToolKit. They did not match! There were two lines in the ToolKit source causing the problem:

```
          DW LIT,$FFFF
L1495     DFB  $C1,$DB
```

QUICKTRACE

relocatable program traces and displays the actual machine operations, while it is running without interfering with those operations. Look at these **FEATURES**:

Single-Step mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.

Trace mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.

Background mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.

QUICKTRACE allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.

Two optional display formats can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.

QUICKTRACE is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.

QUICKTRACE is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.

QUICKTRACE is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.

QUICKTRACE is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program

QUICKTRACE

\$50

Is a trademark of Anthro-Digital, Inc.

Copyright © 1981

Written by John Rogers

See these programs at participating Computerland and other
fine computer stores.

Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202

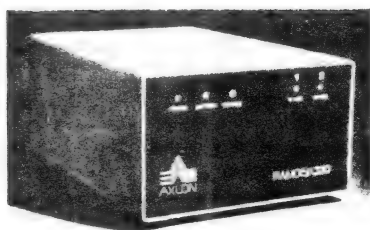
The "DW" directive in ToolKit does not recognize multiple items separated by commas; therefore the ", \$FFFF" was ignored. The following line in the source was "DW \$FFFF". The S-C form ".DA LIT,\$FFFF" does assemble both items, so the \$FFFF constant was duplicated.

The "DFB" directive in ToolKit recognizes multiple items. The conversion I did rendered the line into "L1495 .DA #\$C1,\$DB", so the \$DB item became a 16-bit value. I changed the line to "L1495 .DA #\$C1,\$DB" and all was well.

If you have a large Toolkit source to convert, chances are that you will find one or two more things to change that are not included above. Let me know what you come up with.

As I mentioned earlier, the same general techniques work when you have a LISA file to convert. If you can get the source code on a text file, with all tokens expanded, then you can read it into S-C and begin converting. If you want a challenging assignment, how about writing a program which will read LISA type-B source files and convert them to S-C type-I source files, all automatically!

Supercharge Your APPLE II*



The Axlon RAMDISK™ 320K Memory System for the Apple II and Apple II Plus* provides access speeds never before available. The Axlon memory system is designed to interact with Apple DOS 3.3* and Apple Pascal 1.1* like two standard floppy disk drives while delivering the lightning fast access speeds of RAM memory. This also leaves 32K of RAM for advanced programming techniques. The interface board is slot independent and draws no power from your Apple. The rechargeable battery system built into the unit provides three hours of backup in the event of a power loss. Drop by your local Apple dealer or contact Axlon, Inc. for more information.

Trademark of Apple Computer, Inc.
Pascal is a Trademark of U.C.S.D. Regents

- Plug-in compatibility
- 320K bytes of RAM (200NS) memory designed to function like two 35 track floppy disk drives
- Compatible with Apple DOS 3.3 and Apple Pascal 1.1
- Same size as the Apple Disk II* Drive
- Invisible memory refresh - even with the Apple turned off
- Rechargeable battery system built-in to provide 3 hours of auxiliary power
- Slot independent interface board draws no power from your Apple
- All firmware in static RAM on the interface board
- Includes software for diagnostic, fast load and copy routines, and business applications



170 N. Wolfe Road,
Sunnyvale, CA 94086
(408) 730-0216

Correction to Bob's Fast Screen Scroll.....Jim Church

If you tried the fast scroll from Bob Sander-Cederlof's article "Some Fast Screen Tricks" from the September issue, you might have been surprised. Bob goofed!

He copied characters from line 16 into line 15 before moving line 15 to 14; ditto with lines 8, 7, and 6. This in spite of his special attempt to save lines on the stack. The problem is that he ran the loop backwards from 119 to 0. If you change it to run from 0 up to 119, the scroll works correctly.

Change lines 1410, 1620, and 1630, and add line 1625:

```
1410 SCROLL LDY #0

1620 .2      INY
1625        CMP #120
1630        BCC .1
```

Bob, you are going to get a lot of mail (unless they are asleep)!

RAM/ROM PROGRAM DEVELOPMENT BOARD

\$35.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip. Use a 6116 type RAM chip for program development or just extra memory. Plug in a programmed 2716 EPROM to keep your favorite routines 'on-line'. Maps into \$C000-\$CFFF and \$C800-\$CFFF memory space. Instructions & circuit diagram provided.

The 'MIRROR': Firmware for Apple-Cat

\$29.00

Communications ROM plugs directly into Novation's modem card. Three basic modes: Dumb Terminal, Remote Console & Programmable Modem. Added features include: Printer buffer, Pulse or Tone dialing, true dialtone detection, audible ring detect and ring-back option. Directly supports many 80-column boards (even while printing) and Apple's Comm card commands. (Apple-Cat Hardware differences prevent 100% interchangeability with Comm card.) Includes Hayes-to-AppleCat register equivalences for software conversion. Telephone Software Connection (213-516-9430) has several programs which support the 'MIRROR'.

The 'PERFORMER': Smarts For Your Printer

\$49.00

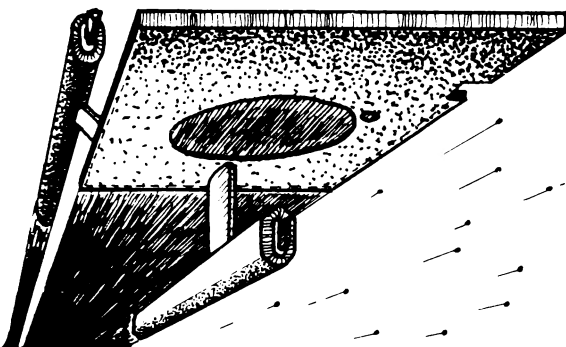
Get the most from your smart printer by adding intelligence to your 'dumb' interface card. The PERFORMER Board plugs into any Apple slot for immediate access (no programs to find and load). Easily select printer fonts and many other features via a user-friendly menu. Replaces manual printer set-up. No need to remember ESC commands. Also provides TEXT and GRAPHICS screen dumps. Compatible with Apple, Tymac, Epson, Microtek and similar 'dumb' Centronics type parallel I/F boards. Specify printer: EPSON MX80 W/Graftrax-80, EPSON MX100, EPSON MX80/MX100 W/Graftrax Plus, NEC 8023A, C.Itoh 8510 (Prowriter), OKI Microline 824/83A W/OKIGRAPH. (OKI Bonus: The PERFORMER Generates ENHANCED and DOUBLE STRIKE Fonts)

Avoid A \$3.00 Shipping/Handling Charge By Mailing Full Payment With Order

R A K - W A R E
41 Ralph Road
West Orange NJ 07052

**** SAY YOU SAW IT IN 'APPLE ASSEMBLY LINE'! ****

**500%
FASTER
THAN
DOS 3.3**



Transform your slow DOS 3.3 into *HyperDOS*

with
HyperDOS creator

A machine-language program that modifies the "in-memory image" of DOS 3.3.

HOW FAST IS HYPERDOS?

LOAD:	DOS 3.3	HyperDOS
Hi-Res picture	13.0 sec.	3.1 sec.
100 sector binary program	24.0 sec.	5.6 sec.
100 sector basic program	24.0 sec.	4.5 sec.

- 1. Use HyperDOS as your HELLO program.**
- 2. INITIALize your new disks with HyperDOS.**
- 3. Replace the old, slow DOS on your disks.**

FOR ONLY

No credit cards **\$19.95**
U.S. funds only

SoftKey Publishing

Dept. AL
P.O. Box 44549
Tacoma, WA 98444

(206) 531-1654

You receive:

HyperDOS CREATOR on disk

Requires 48K Apple II

PLUS

a fully-commented source code listing

a **FREE** copy of **HARDCORE Computing**
the magazine for serious Apple-users

Using USR for a WEEK.....Bob Sander-Cederlof

The "&" and CALL statements are not the only ways to use machine language to enhance the Applesoft Language. USR is a third way, and provides an easy way to return a single value.

How many times have you seen the Applesoft code "PEEK(X) + 256*PEEK(X+1)"? It is used over and over again. What it does is look in memory at X and X+1 for a 16-bit value (stored low-byte first as are most 16-bit values in the 6502 environment). The high byte is multiplied by 256, and the low byte added in. Wouldn't it be nice to have a USR function which would convert a two-byte value directly? This function is sometimes called "WEEK", meaning "Word pEEK" (hence the awful pun in the title above).

When I was in California last week someone categorically and unequivocally assured me that it is impossible to use the USR function with a value of 32768. I tried it with the WEEK function, and it works fine. So much for the assurances! I think his problem was that he followed the instructions in the Applesoft manual, which are somewhat incomplete.

Here is the USR code, set up to run at \$300. However, it is "run-anywhere" code, because there are no internal references. You do have to tell Applesoft where it starts, though. Line 100 in the example shows how to do that. Location 11 and 12 must be set to the low- and high-bytes of the address of the USR code.

	1010	*-----
	1020	* USR (X) = PEEK(X)+256*PEEK(X+1)
	1030	*-----
	1040	.OR \$300 OR WHEREVER YOU WISH
	1050	*-----
0300-	A5 9D	1060 USR LDA \$9D CHECK RANGE
0302-	C9 91	1070 CMP #\$91
0304-	B0 18	1080 BCS .1 ERROR
0306-	20 F2 EB	1090 JSR \$EBF2 CONVERT TO INTEGER IN \$A0,A1
0309-	A5 A0	1100 LDA \$A0 PUT HIGH BYTE AFTER LOW BYTE
030B-	85 A2	1110 STA \$A2
030D-	A0 01	1120 LDY #1
030F-	B1 A1	1130 LDA (\$A1),Y HIGH-ORDER BYTE
0311-	85 9E	1140 STA \$9E HIGH BYTE OF MANTISSA
0313-	88	1150 DEY
0314-	B1 A1	1160 LDA (\$A1),Y LOW-ORDER BYTE
0316-	85 9F	1170 STA \$9F NEXT BYTE OF MANTISSA
0318-	38	1180 SEC SIGN IS POSITIVE
0319-	A2 90	1190 LDX #\$90 EXPONENT 2^16
031B-	4C A0 EB	1200 JMP \$EBA0 FINISH CONVERSION
031E-	4C 99 E1	1210 .1 JMP \$E199 "ILLEGAL QUANTITY" MESSAGE
	1220	*-----

```
100 POKE 11,0: POKE 12,3
105 INPUT X: VTAB PEEK (37): PRINT X": ";
110 PRINT USR (X)" = " PEEK (X) + 256 * PEEK (X + 1)
120 GOTO 105
```

Automatic CATALOG in the Language CardBill Morgan

It has been pointed out to me (loudly!) that I failed to mention the Language Card version of the Macro Assembler in my Automatic CATALOG routine last June. Well, here's what you need to do:

Assemble the patch with an origin of \$DF00. This is a blank page inside the assembler.

BLOAD PATCH.

\$D46D:FF DE In the Language Card version, the Escape code jump table is at \$D467-D482.

BSAVE S-C.ASM.MACRO.LC.MOD,A\$D000,L\$231F

That should take care of it!

APPLE EPROM PLUG™

•**REPLACE** any or all Apple II ROMs with EPROMs with NO Apple modifications. •**CUSTOMIZE** the F8 Autostart ROM to change prompt - change NMI, IRQ and Reset vectors to control any program. Other custom information furnished. •**SECURITY** by putting Apple or program serial number etc., in "free" F8 ROM locations. •**OTHER FEATURES-** Very small, 1 3/16"L x 11/16"W x 5/16"H - no interference with language card in Slot #0 - up to 5" card in other slots without interference.

SPECIAL LIMITED OFFER - Prime EPROM and programming for \$10.00 plus \$1.75 postage with EPROM PLUG purchase.

TO ORDER: Send \$14.95 plus \$1.75 postage for each unit - VISA and Mastercard accepted. TO: **MARTCOMM, INC.**, Dept. B, P.O. Box 74, Mobile, AL 36601 or call (205) 342-7259 after 6 PM CDT.

Another Lower Case Patch for S-C Macro.....Bob Sander-Cederlof

Graeme Scott pointed out another oversight of mine. All lower case characters inside macro definitions are currently converted to upper case, whether or not you want it that way. The following patches will fix it, assuming you have already installed the patches from AAL August 1982 page 28.

Motherboard version: \$275E:BA 31

Language Card version: \$E8AA:06 F3

I found another problem: ".EM" and ".eM" work, but ".em" and ".Em" do not. The following patches make them work too.

Motherboard version:

\$31DB:B9 00 02 C9 60 90 02 29 5F 60
\$2979:20 DB 31

Language Card version:

\$F327:B9 00 02 C9 60 90 02 29 5F 60
\$EAC5:20 27 F3

Writing for AAL.....Bob Sander-Cederlof

More and more of you are expressing interest in contributing articles to this newsletter. Fine with me!

I accept them in almost any form. It is by far the best if any source programs are on disk in S-C format, so I don't have to type them in. Other formats are OK, but more trouble.

I use my own word processor, which accepts standard DOS text files or Applewriter files. If you have a large article, a copy on disk saves a lot of time here.

I receive more articles than I can use, but if yours is as good as you think it is, I will probably print it. I usually spend a lot of time checking the programs and editing the articles before I print them.

Of course, I will return any disks you send.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$15 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$13 postage for other countries. Back issues are available for \$1.50 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)